NC Voter Project

Goal 1

Background 1

Implementation Plan 1

Database Structure 2

Historical Registration Data 3

Current Progress 3

Resources 3

Known Issues and Fixes 4

# 

# Goal

Collect, clean and organize NC voter information

# Background

The NC State Board of Elections uploads current snapshots of their data on <https://s3.amazonaws.com/dl.ncsbe.gov/data/list.html> every few days. The file [ncvoter\_Statewide.zip](https://dl.ncsbe.gov.s3.amazonaws.com/data/ncvoter_Statewide.zip) has the most recent registration information. And the file [ncvhis\_Statewide.zip](https://dl.ncsbe.gov.s3.amazonaws.com/data/ncvhis_Statewide.zip) has the most recent voting records (who vote in what election).

Every time NCSBE updates the snapshots, the files will be overwritten. Hence, one can only get the most up-to-date voter information from the NCSBE website but not previous information (if it ever changed).

We will collect the real-time voter registration data from the ncvoter file, and keep track of its changes over time. The voting history information is easier to deal with; we only need to collect the new voting records every time they are posted and append them to the existing table.

# Implementation Plan

Voter Registration Data

To keep track of voters’ registration data over time, we will use two tables, one for the most recent information, and the other for all the changes that ever occurred.

The NCVoter table will store the most recent registration information for each voter. The ChangeTracker table will store the changes between snapshots for each voter. Besides, we will use one more table FileTracker to store the metadata of each snapshot file.

The main workflow will be as follows.

- Download the snapshot file when it is updated

- For each line in the snapshot file

- Compare that line to the corresponding voter in the NCVoter table

- Represent changes as (field: new value) pairs for fields that are different

- Store the pairs as a JSON blurb in the data field of the ChangeTracker table

- Update the corresponding voter in the NCVoter table

Voting Records

We will use the NCVHis table to store voting records. The FileTracker table will also be used to store the metadata for each snapshot file.

The workflow is straightforward.

- Download the snapshot file when it is updated

- Parse the file and append all rows to the end of the NCVHis table

# Database Structure

FileTracker Table: metadata for the snapshot files.

- etag

- filename

- data\_file\_kind: either voter file or voting history file

- county\_num: unnecessary, to be removed

- created: time when the file is downloaded to local

- file\_status: UNPROCESSED, PROCESSING or PROCESSED

NCVoter Table: most recent voter registration information.

- ncid: the unique identifier for voters

- See the layout information of snapshot files for other fields

ChangeTracker Table: changes to the voter registration data between snapshots.

- op\_code: whether this change creates a voter or modifies an existing voter’s information

- model\_name: unnecessary field, to be removed

- md5\_hash: md5 hash value of this row after it is parsed.

- data: changes occurred to the voter. See **Implementation Plan** for details.

- ncid: ncid of the voter.

- election\_desc: unnecessary field, to be removed.

- file\_tracker: which file does the change come from.

- snapshot\_dt: the publication date of the snapshot file if it is known, or the date when the file is downloaded if its publication date is unknown.

NCVHis Table: voting records (who vote in which election).

- See layout information of the snapshot files for details on the fields.

# Historical Registration Data

As discussed earlier, the [ncvoter\_Statewide.zip](https://dl.ncsbe.gov.s3.amazonaws.com/data/ncvoter_Statewide.zip) file has only the most recent registration data of voters. Consequently, we won’t have data earlier than whenever we start downloading the snapshots. It turns out that this is not completely true.

The NCSBE also keeps historical snapshots of voter registration information at <https://s3.amazonaws.com/dl.ncsbe.gov/data/Snapshots/> or (<http://dl.ncsbe.gov/index.html?prefix=data/Snapshots/>). However, those files are apparently different from the [ncvoter\_Statewide.zip](https://dl.ncsbe.gov.s3.amazonaws.com/data/ncvoter_Statewide.zip) files. The number of historical snapshots is small considering how often the real-time snapshots are updated. Also, the historical files are at least 2 times larger than the real-time snapshots.

A few observations are made into the historical files, which make it possible to use them as historical registration data.

1. The historical files have the same fields as the real-time snapshots, plus a few extra fields such as snapshot date, etc.

2. In each historical file, some ncid’s might appear multiple times. One possibility is that each historical file is a concatenation of multiple real-time snapshots from that period.

In light of those observations, we will collect all the historical registration files first, and then starting collecting the real-time voter snapshots. The code for doing both kinds of snapshots should be the same except for the part of where the files are, so there should be minimal extra work.

# Resources

Database server: Amazon RDS

- Address: ncvoter.c4228hnvlvqh.us-east-1.rds.amazonaws.com

- Database name: ncvoter

- Username: reader (Read access only)

- Password: reader

Amazon EC2 machine:

- Address: 34.204.163.12

# Current Progress

Code for processing the historical snapshots is ready and running.

- voter\_fetch\_snapshot.py: Download and upzip a new historical snapshot file when all local files are processed or under processing.

- voter\_process\_snapshot.py: Process the oldest snapshot file in the FileTracker table, update the database (see Implementation Plan for details), and move to the next available file.

The ingestor has finished snapshot 20120101. The sizes of the tables are

- ncvoter: 8977 MB

- changetracker: 39 GB

# Known Issues and Fixes

1. A few rows in some historical snapshots are longer or short than what they should be.

Temporary fix: A pattern is found. It is the one item or the three items before registration\_dt are missing. Hence, if the length of the row (in terms of items) is 1 or 3 less than what it should be, treat those missing items as empty. If the length is not 1 or 3 less than normal, print the row and wait for input from user.

2. If the ingestor is running, and another one kicks off accidentally, unexpected behavior will happen, resulting in unreliable data in the database.

Temporary fix: When the code starts, check if any file is under processing. If yes, stop immediately.

# Next Steps